A frontal Delaunay quad mesh generator using the L^∞ norm

J.-F. Remacle[2], F. Henrotte[2], T. Carrier Baudouin[2], C. Geuzaine[1], E. Béchet[3], Thibaud Mouton[3], and E. Marchandise[2]

[1] Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium
cgeuzaine@ulg.ac.be

[2] Institute of Mechanics, Materials and Civil Engineering, Universit catholique de Louvain, Avenue Georges-Lematre 4, 1348 Louvain-la-Neuve, Belgium
{jean-francois.remacle,emilie.marchandise,
tristan.carrier,francois.henrotte}@uclouvain.be

[3] Université de Liège, Aerospace and Mechanical Engineering Department, Chemin des Chevreuils, 1, 4000 Liège, Belgium
{eric.bechet, thibaud.mouton}@ulg.ac.be

Summary. A new indirect way of producing all-quad meshes is presented. The method takes advantage of a well known algorithm of the graph theory, namely the Blossom algorithm that computes the minimum cost perfect matching in a graph in polynomial time. Then, the triangulation itself is taylored with the aim of producing right triangles in the domain. This is done using the infinity norm to compute distances in the meshing process. The alignement of the triangles is controlled by a cross field that is defined on the domain. Meshes constructed this way have their points aligned with the cross field direction and their triangles are almost right everywhere. Then, recombination with our Blossom-based approach yields quadrilateral meshes of excellent quality.

Key words: Quadrilateral mesh generation, graph theory, infinity norm

1 Introduction

This paper describes a new methodology for generating meshes with quadrilateral elements (quad meshes). There exist so far essentially two approaches to generate automatically such meshes. With *direct methods*, quadrilaterals are constructed at once, using either advancing front techniques [1] or regular grid-based methods (quadtrees) [2]. *Indirect methods*, on the other hand, rely on an initial triangular mesh and apply merging techniques to recombine the triangles of the initial mesh into quadrangles [3, 4]. Other more sophisticated indirect methods use a mix of advancing front and recombination [5].

In order to motivate our work, let us first explain why standard indirect quadrilateralization methods fail to produce optimal quad meshes. Figure 1-(a) shows a uniform triangular mesh in R^2 with equilateral elements; all elements and all edges are of size a. This mesh can be deemed perfect in the sense that optimality criteria, both in size and shape, are fulfilled. In this case, the Voronoi cell of each vertex $\mathbf{x}$ is an hexagon of area $a^2\sqrt{3}/2$, and the number of points per unit of surface is $2/(a^2\sqrt{3})$. Comparing with a uniform mesh made of right triangles of size a, Figure 1-(b), one sees that the Voronoi cells are now squares of area a^2. Filling R^2 with equilateral triangles requires thus $2/\sqrt{3}$ times more vertices (i.e. about 15% more) than filling the same space with right triangles. So, although quad meshes can be obtained by recombination of any triangular meshes, conventional triangular meshes are not the most appropriate starting point because they are essentially made of (nearly) equilateral triangles and contain therefore about 15% too many vertices. The purpose of this paper is to introduce a method to generate triangular meshes suited for recombination into well-behaved quad meshes.

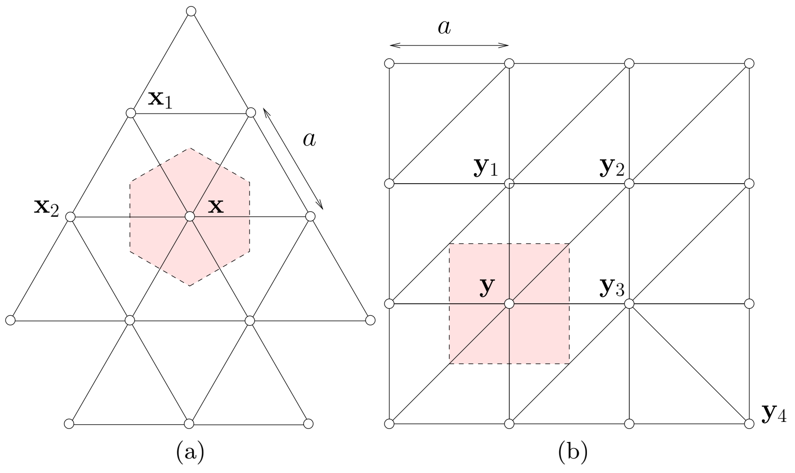


Fig. 1. Voronoi cells of one vertex that belongs either to mesh of a equilateral triangles (a) or of right triangles (b).

The mesh in Figure 1-(b) contains edges of different sizes. For example, $\|\mathbf{y} - \mathbf{y}_2\|_2 = a\sqrt{2}$ whereas $\|\mathbf{y} - \mathbf{y}_1\|_2 = a$. This mesh contains long edges at 45 degrees and short edges aligned with the axis. This explains why mesh (b) contains less points than mesh (a). Yet, long edges will be eliminated by the recombination procedure and the final mesh will be made of quadrilaterals with all edges of size a.

In devising a procedure to generate triangular meshes well-suited for recombination into quadrangles, one should recognize that the optimal size of an edge to be inserted at a point by the Delaunay algorithm depends on its orientation. A first possibility would be to encapsulate this directional information into some kind of anisotropic L^2 metric. This is however not possible as such a metric $\mathcal{M}$ would have to ensure that (See Figure 1-(b))

$$\begin{aligned}(\mathbf{y}_1 - \mathbf{y}_2)^T \mathcal{M}(\mathbf{y}_1 - \mathbf{y}_2) &= (\mathbf{y}_3 - \mathbf{y}_2)^T \mathcal{M}(\mathbf{y}_3 - \mathbf{y}_2) \\ &= (\mathbf{y}_3 - \mathbf{y}_4)^T \mathcal{M}(\mathbf{y}_3 - \mathbf{y}_4) \\ &= (\mathbf{y} - \mathbf{y}_2)^T \mathcal{M}(\mathbf{y} - \mathbf{y}_2),\end{aligned}$$

which is clearly impossible. This suffices to conclude that standard metric-based triangulation algorithms are unable to produce meshes made exclusively of right triangles.

The approach proposed in this paper is based on the following observation. If distances between points are measured in the L^∞-norm, the triangular elements of Figure 1-(a) are no longer equilateral: $\|\mathbf{x} - \mathbf{x}_2\|_\infty = a$ and $\|\mathbf{x} - \mathbf{x}_1\|_\infty = a\sqrt{3}/2$. On the other hand, the elements of Figure 1-(b), which are right triangles in the L^2 norm, are equilateral in the L^∞-norm: $\|\mathbf{y} - \mathbf{y}_1\|_\infty = \|\mathbf{y} - \mathbf{y}_2\|_\infty = a$.

On this basis, the frontal Delaunay algorithm can be adapted to work in the L^∞-norm so as to produce triangular meshes with the right number of nodes and triangles suitably shaped for producing high quality quadrilaterals after recombination.

The paper is divided in three parts. In the first part, we make use of a famous algorithm of the theory of graphs: the Blossom algorithm, proposed by Edmonds in 1965 [6, 7], which allows to find the minimal cost perfect matching of a given graph. While classical triangle merge procedures [3, 4] are based on some kind of heuristics that allow to find which pairs of triangles forming good quadrangles after recombination, the new method has some clear advantages: (i) it provides a mesh that is guaranteed to be quadrilateral only, (ii) it is optimal in a certain way and (iii) it is fast.

Then, a method to build the so-called “cross fields” [8] is presented. The cross fields represent at each point of the domain the preferred orientations of the quadrilateral mesh. In the finite element community, it is usually appreciated that quadrilateral elements have orientations parallel to the domain boundaries.

Finally, the mesh generation procedure is described in detail. A frontal Delaunay approach inspired by [9] is proposed for determining the successive position of new points. Frontal meshers usually insert a point in the mesh so as to form an equilateral triangle (in L^2-norm). Here, we also aim at generating an equilateral triangle, yet in the sense of the local L^∞-norm aligned with the cross field. Meshes constructed this way have their points aligned with the cross field direction and their triangles are almost right everywhere. Then, recombination with our Blossom-quad approach [10] yields quadrilateral meshes of excellent quality.

In all meshes that are presented as results, the quality of the quadrangular meshes are evaluated by computing the quality η of every quadrangle as follows:

$$\eta = \max\left(1 - \frac{2}{\pi}\max_k\left(\left|\frac{\pi}{2} - \alpha_k\right|\right), 0\right), \qquad (1)$$

where $\alpha_k, k = 1,..,4$ are the four angles of the quadrilateral. This quality measure is $\eta = 1$ if the element is a perfect quadrilateral and is $\eta = 0$ if one of those angles is either ≤ 0 or $\geq \pi$. The average quality of elements is noted $\bar{\eta}$ and the worst element quality is noted η_w.

2 Triangle merging using the Blossom algorithm

The idea of the Blossom algorithms is to build a specific weighted graph $G(V, E, c)$ from the triangle adjacencies in a given mesh $\mathcal{T}_0$. Fig. 2 shows a simple triangular mesh with its graph G. Here, V is the set of graph vertices, E the set of graph edges and $c(E)$ an graph-edge-based cost function. As can be seen every vertex of the graph is a triangle t_i of the mesh and every edge of the graph is an internal edge e_{ij} of the mesh that connects two neighboring triangles t_i and t_j. We aim at finding a subset of edges that forms a perfect matching, i.e. that makes pairs of triangles, leaving no triangle alone.

2.1 Blossom: a minimum cost perfect matching algorithm

Let us consider $G(V, E, c)$ now such an undirected weighted graph. A *matching* is a subset $E' \subseteq E$ such that each node of V has at most one incident edge in E'. A matching is said to be perfect if each node of V has exactly one incident edge in E'. As a consequence, a perfect matching contains exactly $|E'| = |V|/2$ edges. This means that a perfect matching can only be found for graphs with an even number of vertices. A matching is optimum if $c(E')$ is minimum among all possible perfect matchings.

In 1965, Edmonds [11, 6] invented the *Blossom algorithm* that solves the problem of optimum perfect matching in polynomial time. A straightforward implementation of Edmonds's algorithm requires $\mathcal{O}(|V|^2|E|)$ operations. Since then, the worst-case complexity of the Blossom algorithm has been steadily improving. Both Lawler [12] and Gabow [13] achieved a running time of $\mathcal{O}(|V|^3)$, Galil, Micali and Gabow [14] improved it to $\mathcal{O}(|V||E|\log(|V|))$. The current best known result in terms of $|V|$ and $|E|$ is $\mathcal{O}(|V|(|E| + \log|V|))$ [15].

There is also a long history of computer implementations of the Blossom algorithm, starting with the Blossom I code of Edmonds, Johnson and Lockhart [7]. In this paper, our implementation makes use of the Blossom IV code of Cook and Rohe [16][4] that has been considered for several years as the fastest available implementation of the Blossom algorithm.

[4] Computer code available at `http://www2.isye.gatech.edu/ wcook/blossom4`.

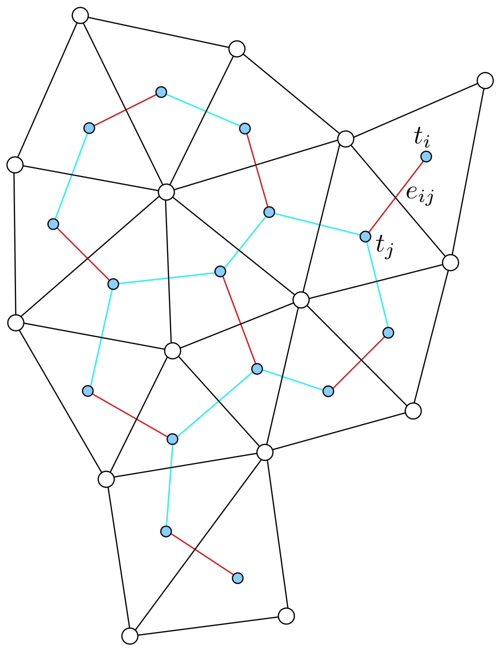


Fig. 2. A mesh (in black) and its graph (in cyan and red). The cyan points are the V graph vertices, the cyan and red lines the set of graph edges E and the subset of red edges forms a perfect matching.

2.2 Optimal triangle merging

In term of what has just been defined, the subset E' of edges that have been used for triangle merging in the approach of [4] is a matching that is very rarely a perfect matching. The one of [3] is usually a perfect matching, but not necessarily the optimal one.

Here, we propose a new indirect approach to quadrilateral meshing that takes advantage of the Blossom algorithm of Edmonds. To this end we apply the Blossom IV algorithm to the graph of the mesh. We intend to find the optimum perfect matching with respect to the following total cost function

$$c = \sum_{e \in E'} (1 - \eta(q_{ij})), \tag{2}$$

that is, the sum of all elementary cost functions (or “badnesses”) q_{ij} of the quadrilaterals that result in the merging of the edges of the perfect matching E'. Such a cost function can be related to any mesh quality measure [17].

An obvious requirement for the final mesh to be quadrilateral only is that the initial triangular mesh contains an even number of triangles (i.e., an even number of graph vertices). Euler’s formula for planar triangulations states that the number of triangles in the mesh is

$$n_t = 2(n_v - 1) - n_v^b, \tag{3}$$

where n_v^b is the number of mesh nodes on its boundary. So, the number of mesh points on the boundary n_v^b should be even.

If for some graphs it is possible to find different perfect matchings, there is in general no guarantee that even one single perfect matching exists in a given graph. The general problem of counting the number of perfect matchings in a general graph is #P-complete[5]. In other words, there is no hope to find the number of perfect matchings in a general graph (however, there is a way to find out, in polynomial time, wether a perfect matching exists by detecting a breakdown in the Blossom algorithm). More details about Blossom, matchings and graphs can be found in [10], where we describe a way to ensure that the graph of the mesh always has a perfect matching.

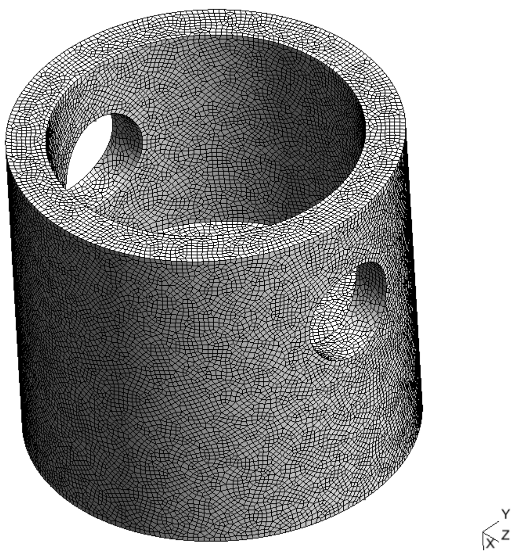


Fig. 3. Mesh of the Piston created from a combination of triangle using the Blossom-quad algorithm.

Fig. 3 shows an example of the use of the Blossom algorithm for recombining the triangles together with the optimization procedure that is described in [10]. The mesh is composed of 39,386 quadrilateral elements that are of average quality $\bar{\eta} = 0.78$, which is good but not great. It is indeed clear that

[5] Sharp p-complete, i.e. much harder than NP-complete.

elements have no preferred orientation. The mesh is composed of patches of quadrilaterals that have random orientations. In the next section we will develop a new method for inserting points in the domain in a way such that Blossom will be able to provide an optimal quad-mesh that is oriented correctly.

3 Cross fields

In what follows, we consider that we have a first triangulation of a surface $\mathcal{T}_0$ in $\mathcal{R}^3$ and that we have build a parametrization $\mathbf{x}(\mathbf{u})$ that maps every point of the 3D surface mesh to a point in a parametric space in $\mathcal{R}^2$. The parametrized surface mesh is denotes $\mathcal{T}_0'$. Moreover, we are able to compute the derivatives of the mapping in order to build the mesh metric :

$$\mathbf{M} = \mathbf{x}_{,\mathbf{u}}^T \mathbf{x}_{,\mathbf{u}} = \begin{bmatrix} \mathbf{x}_{,u} \cdot \mathbf{x}_{,u} & \mathbf{x}_{,u} \cdot \mathbf{x}_{,v} \\ \mathbf{x}_{,v} \cdot \mathbf{x}_{,u} & \mathbf{x}_{,v} \cdot \mathbf{x}_{,v} \end{bmatrix}. \quad (4)$$

A cross field $\theta(\mathbf{u})$ is supposed to give enough information to orient a local system of axis at each point $\mathbf{u}$ in the parameter plane. The edges of the quadrilaterals generated around $\mathbf{u}$ should then be aligned with the cross field. Figure 4 shows an annular domain, the cross field and the resulting quad mesh. The computer graphics community has already been confronted to the issue

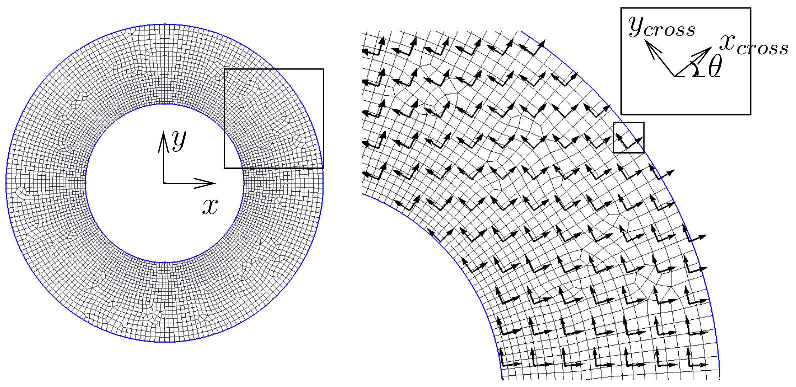


Fig. 4. Mesh of annular domain and a zoom on the cross field.

of computing "cross fields" in the context of (global) surface parametrization [18, 19]. Cross fields can be based on principal directions of curvature of the surface [8].

Here, we consider an *ad hoc* approach based on the following criteria:

- The cross field should be computed automatically;
- Mesh directions should be parallel to the boundaries of the domain at the vicinity of those boundaries;
- The cross field should be smooth.

In order to fulfill those constraints, we have chosen to compute θ using a boundary value problem. The value of θ is fixed at the boundary of the domain and is propagated inside the domain using an elliptic PDE.

The angular oriention of a cross being defined up to the multiples of $\pi/2$, it cannot be represented univocally by the orientation of one branch of the cross. The complex valued function

$$\alpha(\mathbf{u}) = a(\mathbf{u}) + ib(\mathbf{u}) = e^{4i\theta(\mathbf{u})}$$

however offers an univocal representation, as it takes one same value for the directions of the 4 branches of a local cross.

A first triangular mesh $\mathcal{T}_0$ is generated using any available algorithm. If a parametrization of the surface needs to be computed, then we use the same mesh as the one that has been used for computing the parametrization. Two Laplace equations with Dirichlet boundary conditions are then solved for each surface of the mesh in order to compute the real part $a(\mathbf{u}) = \cos 4\theta$ and the imaginary part $b(\mathbf{u}) = \sin 4\theta$ of α:

$$\begin{aligned} \nabla^2 a = 0, \quad \nabla^2 b = 0 \quad &\text{on} \;\; \mathcal{T}_0', \\ a = \cos(4\theta_b), \quad b = \sin(4\theta_b) \quad &\text{on} \;\; \partial\mathcal{T}_0', \end{aligned} \tag{5}$$

where θ_b is the angle between the normal to the boundary and the coordinate axis. Then, we have to supply the boundary conditions $\bar{a}(\mathbf{u})$ and $\bar{b}(\mathbf{u})$ ensuring that θ is aligned with $\partial\mathcal{S}'$. After solving, the cross field is represented by

$$\theta(\mathbf{u}) = \frac{1}{4}\text{atan2}(b(\mathbf{u}), a(\mathbf{u})),$$

where $\text{atan2}(b, a)$ is the 4-quadrant inverse tangent.

4 Triangulation in the L^∞-norm

In this section, standard geometrical notions usually defined in the L^2-norm are extended in the L^∞ norm.

4.1 Distances and norms

In the R^2 plane, the distance between two points $\mathbf{x}_1(x_1, y_1)$ and $\mathbf{x}_2(x_2, y_2)$ is usually based on the Euclidean norm (L^2-norm). Other distances can be defined however, based on other norms:

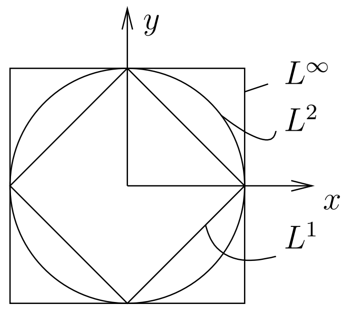


Fig. 5. Illustration of the unit circle in different norms $\|\mathbf{x}\|_p$

- The L^1-norm distance $\|\mathbf{x}_2 - \mathbf{x}_1\|_1 = |x_2 - x_1| + |y_2 - y_1|$,
- The L^2-norm distance $\|\mathbf{x}_2 - \mathbf{x}_1\|_2 = (|x_2 - x_1|^2 + |y_2 - y_1|^2)^{1/2}$,
- The L^p-norm distance $\|\mathbf{x}_2 - \mathbf{x}_1\|_p = (|x_2 - x_1|^p + |y_2 - y_1|^p)^{1/p}$,
- The L^∞-norm distance $\|\mathbf{x}_2 - \mathbf{x}_1\|_\infty = \lim_{p\to\infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p = \max(|x_2 - x_1|, |y_2 - y_1|)$.

Figure 5 shows the unit circle in different norms. One important thing to remark is that only the L^2-norm is rotation invariant. The L^∞-norm depends this on the local orientation of the coordinate axes.

In order to simplify the notations, we consider in what follows that (x, y) are the local coordinates aligned with the cross field (x_{cross}, y_{cross}) (see for example the cross field in Fig. 4).

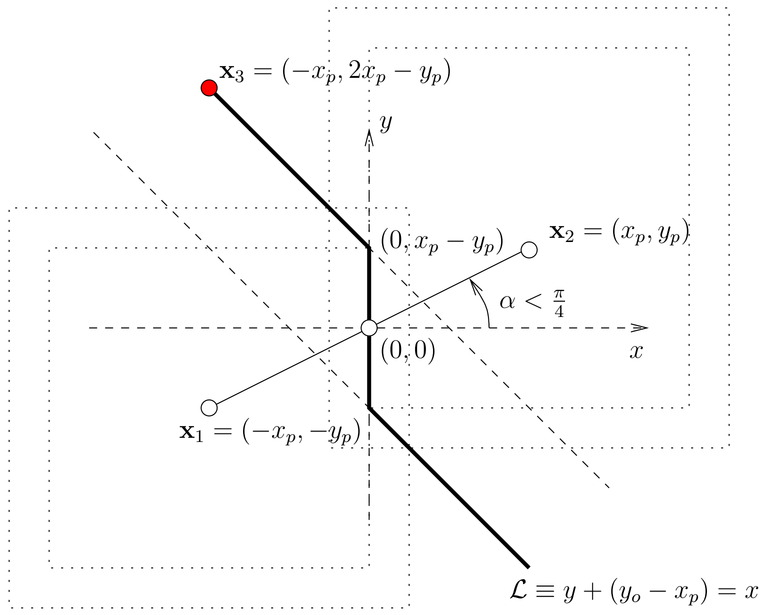
4.2 Bisectors in the L^∞-norm

The perpendicular bisector, or bisector of the segment delimited by the points $\mathbf{x}_1 = (-x_p, -y_p)$ and $\mathbf{x}_2 = (x_p, y_p)$ is by definition the set of points $\mathbf{x} = (x, y)$ equidistant to $\mathbf{x}_1$ and $\mathbf{x}_2$. In the L^2-norm, it is the union of all 2 by 2 intersections of circles centered at $\mathbf{x}_1$ and $\mathbf{x}_2$ and having the same radius. Those intersections are each times two points and their union form a straigth line. In the L^∞-norm, the circles have the geometric appearance of squares and their intersections 2 by 2 are either 2 points or a segment. The bisector is then a broken line (see Fig. 6) in general but it can also form a diabolo-shaped region whenever the considered segment is aligned with an axis ($x_1 = x_2$ or $y_1 = y_2$).

It is assumed, without loss of generality, that $x_p \geq y_p$. The bisector of the segment in the L^∞-norm is the set

$$\mathcal{L} = \{\mathbf{x} = (x, y), \max(|x - x_p|, |y - y_p|) = \max(|x + x_p|, |y + y_p|)\}.$$

The vertical segment of Figure 6 that passes through the origin $(0, 0)$ is the intersection of the L^∞-circles of L^∞-radius x_p centered at $\mathbf{x}_1$ and $\mathbf{x}_2$. It belongs



$\mathbf{x}_3 = (-x_p, 2x_p - y_p)$
y
$(0, x_p - y_p)$
$\mathbf{x}_2 = (x_p, y_p)$
$\alpha < \frac{\pi}{4}$
$(0, 0)$
x
$\mathbf{x}_1 = (-x_p, -y_p)$
$\mathcal{L} \equiv y + (y_o - x_p) = x$

Fig. 6. Bisector of two points $\mathbf{x}_1 = (-x_p, -y_p)$ and $\mathbf{x}_2 = (x_p, y_p)$ using the L^∞-norm. The dottes squares are the L_∞ circles.

thus to the bisector. Increasing now the radius progressively, the intersection of the two L^∞-circles is a pair of points forming two half lines oriented at $3\pi/4$ and starting at $(0, x_p - y_p)$ and $(0, -x_p + y_p)$ respectively. The equation of the bisector $\mathcal{L}$ is then:

$$\mathcal{L} \equiv y + (y_p - x_p) = x \tag{6}$$

There exists an ambiguity when $y_p = 0$. In this case, the bisector contains 2D regions of the plane. It is assumed in what follows that points are in general position, i.e. that there does not exist two points that share either the same x or y coordinate.

4.3 The equilateral triangle using the L^∞-norm

A triangle $T(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ is equilateral in the L^∞-norm if

$$\|\mathbf{x}_2 - \mathbf{x}_1\|_\infty = \|\mathbf{x}_3 - \mathbf{x}_1\|_\infty = \|\mathbf{x}_3 - \mathbf{x}_2\|_\infty.$$

It is possible to build such a triangle starting from Figure 6. We take again $\mathbf{x}_1 = (-x_p, -y_p)$ and $\mathbf{x}_2 = (x_p, y_p)$ and look for a point on the bisector of $\mathbf{x}_1\mathbf{x}_2$ located at a distance $2x_p$ from the endpoints of the segment. This point is $\mathbf{x}_3 = (-x_p, 2x_p - y_p)$.

4.4 Circumcenter, circumradius and circumsquare in the L^∞-norm

Consider a triangle $T_i(\mathbf{x}_1,\, \mathbf{x}_2,\, \mathbf{x}_3)$. Its circumcenter $\mathbf{x}_c = (x_c, y_c)$ in the L^∞-norm verifies

$$\|\mathbf{x}_1 - \mathbf{x}_c\|_\infty = \|\mathbf{x}_2 - \mathbf{x}_c\|_\infty = \|\mathbf{x}_3 - \mathbf{x}_c\|_\infty .$$

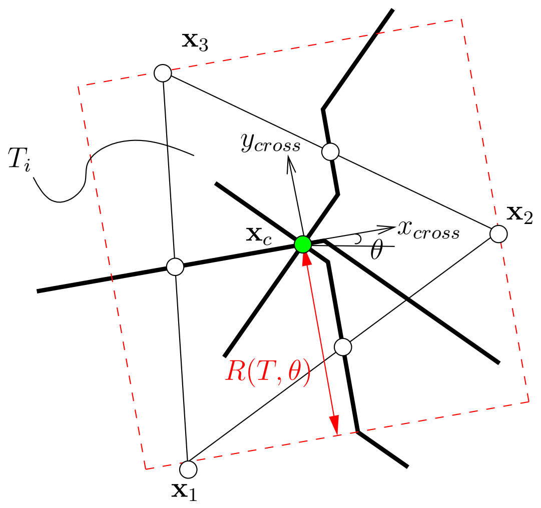


Fig. 7. Circumcenter $\mathbf{x}_c$ and circumradius $R_\infty(T,\theta)$ of a triangle T_i using the L^∞-norm. The circumsquare is the red dotted square.

The L^∞-circumcenter of the triangle is located at the intersection of the L^∞-perpendicular bisectors.

The circumcircle in the L^∞-norm (also called circumsquare), is the smallest square centered at the circumcenter that encloses the triangle, Figure 7. The circumradius $R_\infty(T,\theta)$ is the distance in the L^∞-norm between the circumcenter and anyone of the three vertices. It is given by:

$$R_\infty(T,\theta) = \frac{1}{2} \max\left((\max(x_1,x_2,x_3) - \min(x_1,x_2,x_3), (\max(y_1,y_2,y_3) - \min(y_1,y_2,y_3)\right) . \tag{7}$$

One interesting fact is that the computation of circumcenters and circumradii in the L^∞-norm is a very stable numerical operation.

5 A frontal-Delaunay mesher in the L^∞-norm

Let us recall briefly the pros and cons of the two main approaches for mesh generation. Advancing front techniques start from the discretization of the boundary (edges in 2D). The set of edges of the boundary discretization is called the front. A particular edge of this front is selected and a new triangle is formed with this edge as its base and the front is updated accordingly. The algorithm advances in the domain until theL^∞ front is emptied and the domain fully covered by triangles. The main advantage of advancing front techniques is that they generate points and triangles at the same time, which makes it possible to build optimum triangles, e.g. equilateral triangles in our case. The main drawback of the method is that parts of the front advance independently, leading to possible clashes when they meet.

Delaunay-based mesh generation techniques are more robust because a valid mesh exists at each stage of the mesh generation process. Yet, inserting a point using the Delaunay kernel [20] requires the creation and the deletion of a number of triangles, so that there is less control on the element shapes than in the case of advancing front techniques.

The frontal Delaunay approach makes the best of both techniques. As it is based on a Delaunay kernel, a valid mesh is maintained at each stage of the process. Yet, some kind of front is defined in the triangulation and points are inserted in a frontal manner. The process stops when every element of the mesh has the right size according to the size field $\delta(\mathbf{x})$.

The ideas of the new frontal-quad algorithm are inspired by the frontal Delaunay approach of [9].

In what follows, we consider parametric surfaces that have a conformal parametrization i.e parametrization that conserv angles. Withing this perspective, isotropic meshes on the parameter plane result in isotropic meshes in the 3D space. This hypothesis of conformality may seem over restrictive: for example, the usual parametrization of a spherical surface using spherical coordinates is not conformal. Nevertheless, in this paper, we use reparametrization techniques that allow to build conformal mappings [19, 21]. Moreover, the technique that is presented here can be extended to anisotropic quadrilateral mesh generation.

Consider a surface mesh $\mathcal{T}_0$ for which me have computed a discrete conformal parametrization $\mathbf{u}(\mathbf{x})$ (the parametrized mesh is $\mathcal{T}_0'$) and a given mesh size field $\delta(\mathbf{x})$. Consider also that we have computed on $\mathcal{T}_0'$ a cross field $\theta(\mathbf{u})$ from (5). An new *delquad* mesh $\mathcal{T}_1'$ is constructed in the parameter plane that contains the parametrized points of the boundary edges of $\mathcal{T}_0$. Let us define an adimensional L^∞-meshsize

$$h_i = \frac{R_\infty(T_i, \theta(\mathbf{u}))}{\delta(\mathbf{x}(\mathbf{u}))|\det M(\mathbf{u})|^{1/4}} = \frac{R_\infty(T_i, \theta(\mathbf{u}))}{\delta'(\mathbf{u})}$$

for each triangle T_i of $\mathcal{T}_0$. Quantities $\theta(\mathbf{u})$, $M(\mathbf{u})$ and the size function $\delta(\mathbf{x}(\mathbf{u}))$ are evaluated at the usual (L^2)-centroid of the triangle. Triangles are then classified into three categories

1. A triangle is *resolved* if $h_i \leq h_{\max}$;
2. A triangle is *active* if $h_i > h_{\max}$ and, either one of its three neighbors is resolved or one of its sides is on the boundary of the domain;

3. A triangle is *waiting* if it is neither resolved nor active.

We choose $h_{\max} = 4/3$. This choice is standard in the domain of mesh generation [22]. Figure 8 is an illustration of the way triangles are classified in the algorithm. The front is defined as the set of active triangles. Active triangles

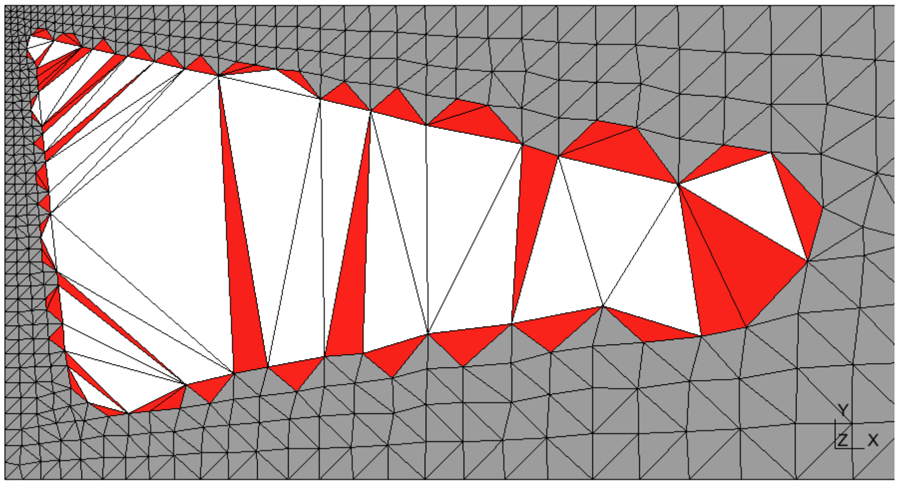


Fig. 8. Illustration of the frontal algorithm with resolved (grey), active (red) and waiting (white) triangles.

are sorted with respect to h_i. Front edges are therefore defined as those edges separating active and resolved triangles.

The frontal algorithm inserts a new point so as to form an optimal triangle with the edge corresponding to the largest active triangle. Consider the edge $\mathbf{x}_2\mathbf{x}_3$ in Figure 9 and assume it corresponds to the largest active triangle of the mesh (the red triangle on Figure 9). For the discussion, the coordinate system has been centered at the mid-edge point $\mathbf{x}_m = \frac{1}{2}(\mathbf{x}_2 + \mathbf{x}_3)$ and aligned with the local cross field, this can be achieved by a translation and a rotation of angle $\theta(\mathbf{x}_m)$.

We choose to position of the new point $\mathbf{x}_n$ along the L^∞-perpendicular bisector $\mathcal{L}$ of $\mathbf{x}_2\mathbf{x}_3$. The exact position of the new point $\mathbf{x}_n$ will be chosen in order to fullfill the size criterion $\delta(\mathbf{x}_m)$.

In order to create a new triangle $T_i(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_n)$ of size $R_\infty(T_i, \theta) = \delta'(\mathbf{x}_m)$, we position $\mathbf{x}_n$ at the intersection of $\mathcal{L}$ with the square C_n of side $\delta'(\mathbf{x}_m)$ passing through points $\mathbf{x}_2$ and $\mathbf{x}_3$ (see Figure 9).

The following considerations should be made.

- The new point should not be beyond $\mathbf{x}_c$, the center of the circumsquare of the active triangle (see Figure 9) as this would create a triangle with

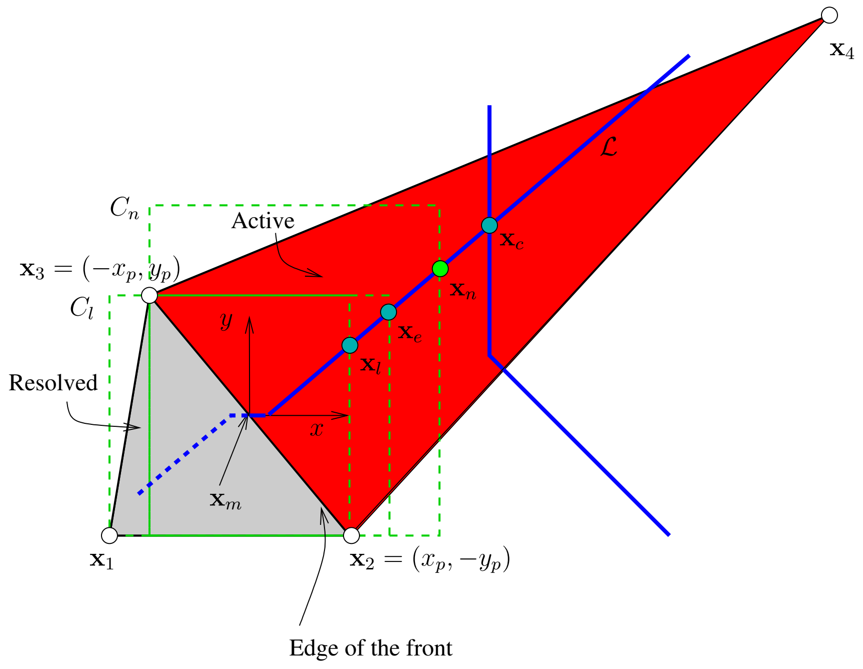


Fig. 9. Illustration of the point insertion algorithm.

a small edge $\mathbf{x}_n\mathbf{x}_4$. Note that this limit case corresponds to a classical point insertion scheme where new points are inserted at the center of the circumcircle of the worst triangle, yet in the L^∞-norm in this case.

- The new point should not be placed below $\mathbf{x}_l$ where $\mathbf{x}_l$ is the intersection of the ∞-perpendicular bisector of $\mathbf{x}_2\mathbf{x}_3$ and the circumsquare C_l of the resolved triangle $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. Inserting a point into C_l would make the resolved triangle invalid by means of the Delaunay criterion.
- If $\delta'(\mathbf{x}_m) = \|\mathbf{x}_3 - \mathbf{x}_2\|_\infty$, then the optimal point is $\mathbf{x}_n = \mathbf{x}_e$. It correspond to the largest triangle $T_i(\mathbf{x}_e, \mathbf{x}_2, \mathbf{x}_3)$ that verifies $R_\infty(T_i, \theta) = \delta'(\mathbf{x}_m)$.

The position of the optimal point is computed as follow:

$$\mathbf{x}_n = \mathbf{x}_e + t(\mathbf{x}_c - \mathbf{x}_e) \tag{8}$$

with

$$t = \min\left(\max\left(1, \frac{\|\mathbf{x}_3 - \mathbf{x}_2\|_\infty - \delta'(\mathbf{x}_m)}{\|\mathbf{x}_3 - \mathbf{x}_2\|_\infty - \|\mathbf{x}_c - \mathbf{x}_2\|_\infty}\right), -\frac{\|\mathbf{x}_l - \mathbf{x}_e\|_2}{\|\mathbf{x}_c - \mathbf{x}_e\|_2}\right), \tag{9}$$

and $\mathbf{x}_c, \mathbf{x}_e$ and $\mathbf{x}_l$ are computed from the equation of the bisector $\mathcal{L}$ (6) :

$$\mathbf{x}_c = \left(\frac{1}{2}(x_4 - x_p), \frac{1}{2}(x_4 + x_p) - y_p\right),$$

$$\mathbf{x}_e = (\delta'(\mathbf{x}_m) - x_p + y_p, \delta'(\mathbf{x}_m)) \quad \text{and} \quad \mathbf{x}_l = (\delta'(\mathbf{x}_m), \delta'(\mathbf{x}_m) + x_p - y_p)\,.$$

Another important ingredient of the advancing front strategy is the fact that the fronts should be updated layer by layer. A initial front is created with the edges of the 1D discretization. The algorithm inserts points until every edge of the active front have been treated. Then other fronts are created and emptied until no active triangle is left in the mesh.

6 Examples

6.1 Piston

As a first example, let us apply the new algorithm to the geometry of the same piston of Figure 3. The result of the advancing front delaunay quad mesher is shown in Figure 10. The new mesh is close to be perfect. The quadrilateral mesh has been automatically generated with the new algorithm, the only control parameter being a constant mesh size field. The mesh is composed of $31,979$ quads and has been generated in about 15 seconds. Compared with the $39,386$ quads of the mesh shown in Figure 3, this mesh has about 19% less nodes. This number is close to the theoretical value $1 - \sqrt{3}/2 = 0.134$.

The average element quality is now $\bar{\eta} = 0.93$ and the worst element is of quality 0.39, which can be considered as very good. Moreover, 92% of the nodes have 4 adjacent quadrangles, which is also very good.

6.2 Falcon aircraft

As a second example, let us consider the Falcon aircraft of Figure 11. The mesh size field is composed of a uniform bulk size field $\delta_b = 0.1$ and of line and point sources positioned at strategic zones of the aircraft.

The resulting mesh is presented on Figure 11. The mesh is composed of $53,297$ quadrangles. The total time for the surface meshing was 22 seconds. The average and worst quality of the mesh are $\bar{\eta} = 0.86$ and $\eta_w = 0.17$ which can be considered as excellent.

7 Conclusion

A new method for automatic quad-meshing of surfaces has been proposed. The new algorithm uses distances in the L^∞ norm as a base for the insertion of new points in the mesh and the generation of edges of the right size in this specific norm.

Perspectives of this approach are numerous. The automatic generation of hex-dominant meshes remains a challenge in the community of finite elements in general. The extension of the new Delquad approach to a Delhex algorithm

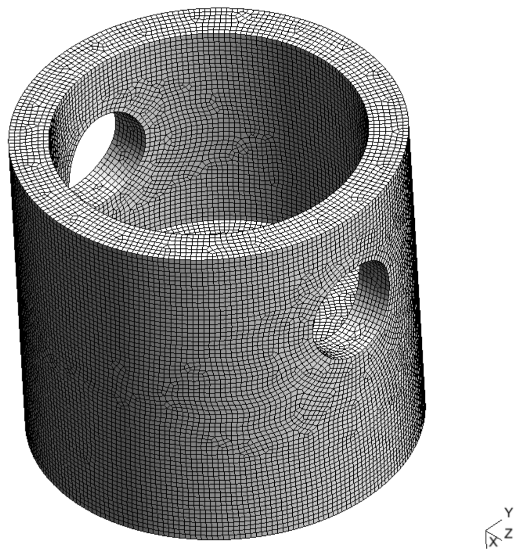


Fig. 10. Mesh of the Piston computed with the presented frontal *delquad* mesh algorithm.

that would generate 3D tetrahedral meshes that have the right number of points and the right orientation to be recombined optimally into hexaedra is a natural sequel to this work.

Acknowledgements

This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074.

References

1. T. D. Blacker and M. B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.
2. P.J. Frey and L. Marechal. Fast adaptive quadtree mesh generation. In *in: Proceedings of the Seventh International Meshing Roundtable.* Citeseer, 1998.
3. C. K. Lee and S. H. Lo. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures*, 52:847–857, 1994.
4. H. Borouchaki and P.J. Frey. Adaptive triangular–quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 45(5):915–934, 1998.

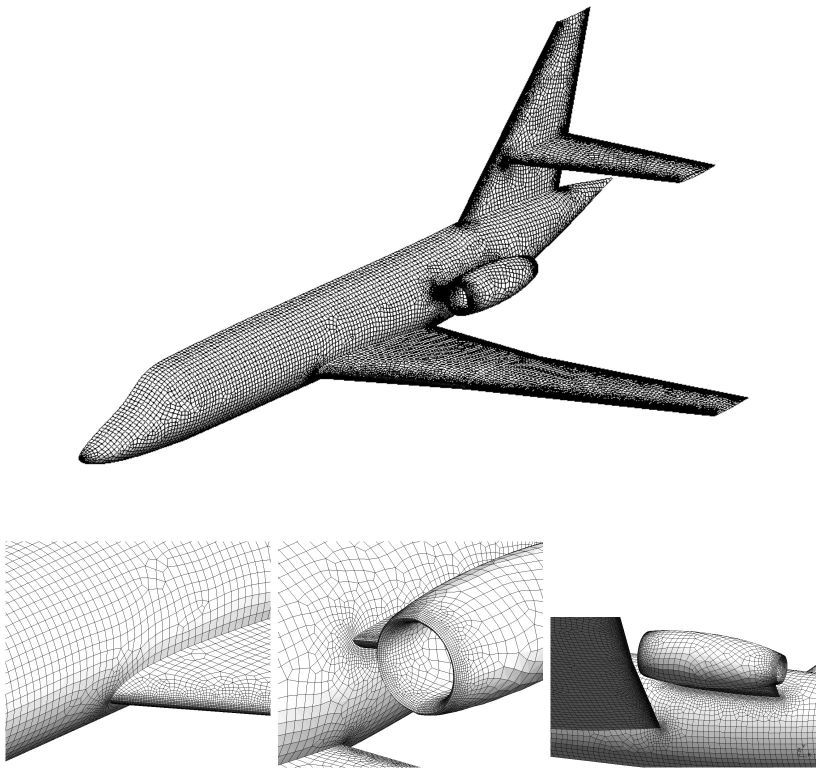


Fig. 11. Final quadrilateral surface mesh of the Falcon aircraft.

5. S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal. Q-morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 9:1317–1340, 1999.
6. J. Edmonds. Maximum matching and a polyhedron with 0-1 vertices. *J. of Research at the National Bureau of Standards*, 69B(125–130), 1965.
7. J. Edmonds, E. L. Johnson, and S. C. Lockhart. Blossom I: A computer code for the matching problem. IBM T. J. Watson J. Edmonds, E. L. Johnson, and S. C. Lockhart. IBM T. IBM T.J. Watson Research Center, Yorktown Heights, New York, 1969.
8. B. Lévy and Y. Liu. Lp centroidal voronoi tesselation and its applications. In *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 2010.
9. S. Rebay. Efficient unstructured mesh generation by means of delaunay triangulation and bowyer-watson algorithm. *Journal of Computational Physics*, 106(1):125–138, 1993.
10. J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, and C. Geuzaine. Blossom-quad: a non-uniform quadrilateral mesh generator using a

minimum cost perfect matching algorithm. *International Journal for Numerical Methods in Engineering*, 2011. submitted.
11. J Edmonds. Paths, trees, and flowers. *Canad. J. Math*, 17:449–467, 1965.
12. E. L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart, and Winston, New York, NY, 1976.
13. H. Gabow. *Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs.* PhD thesis, Stanford University, 1973.
14. H. Gabow, Z. Galil, and S. Micali. An o(ev log v) algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Computing,*, 15(120–130), 1986.
15. H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In 434-443, editor, *In Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms,*, 1990.
16. W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing,*, 11(2)(138–148), 1999.
17. P. P. Pébay. Planar quadrangle quality measures. *Engineering with Computers*, 20(2):157–173, 2004.
18. D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–10, New York, NY, USA, 2009. ACM.
19. B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002.
20. D. F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.
21. E. Marchandise, C.C. de Wiart, WG Vos, C. Geuzaine, and J.F. Remacle. High-quality surface remeshing using harmonic maps–part ii: Surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering*, 86:1303–1321, 2011.
22. P.J. Frey and P.-L. George. *Mesh Generation - Application To Finite Elements.* Wiley, 2008.